Appendix A: Application Security Guidelines

Scope of Application Level Security

1. For the purposes of this document, Application Level security refers to the security requirements that must be met by the application in order to control access to and protect data maintained by the application.

2. The document presumes there is additional security outside of the application that allows system administrators to control access to the State's computers, network, and databases. This document is designed to supplement, not to supercede State security policies and standards published at http://www.doit.state.ct.us.

3. The document presumes that the current methods of securing applications vis-à-vis the state firewall are sufficient and necessary to provide protection.

4. The document presumes that both procured systems and internally developed systems will adhere to the standards proposed.

5. This document includes standards sufficient to meet the requirements of the 1998 proposed regulations. When systems are designed/procured a full analysis of any current HIPAA requirements will have to be made.

Planning for Security

Application level security must be planned from the start of any project and should be reviewed during any upgrade, enhancement or major maintenance activity to existing projects. Security planning should include an analysis of security requirements, which may include the following topics:

1. Review of current and future business goals of the system.
2. Analysis of system assets relative to the value and sensitivity of the data.
3. Analysis of potential risks to the assets.
4. Review of regulatory requirements regarding both security and privacy of data.
5. Analysis of both internal and external threats to security.
6. Develop the goals of your security system.

As part of the analysis of risk, the project team should consider:

1. Fraud
2. Theft
3. Destruction
4. Breach of privacy
5. Denial of service

The final goals of your security system should include:

1. Confidentiality of the data
2. Authentication of the data
3. Integrity of the data
4. Non-repudiation of the data

5. Availability of the data
6. Auditing of the data

Security application design guidelines

The following guidelines are designed to present a high-level description of architectural design patterns and best practices that will aide the practitioner in implementing functional and adaptable security mechanisms.

Single point of entry

Design applications with a single entry point where all appropriate security features are implemented.

Benefits:

Easier security audits
Increased adaptability
Eliminate duplicate code
Prevent back doors

Implications:

A mechanism to direct users to different points within an application after passing through the single entry point may need to be developed.

Secure Access Layer

Isolate low-level security functions (operating system, network, authentication methods, and encryption) in a single layer. Augment existing low-level security functions within this layer to meet application security requirements.

Benefits:

Easier security audits
Increased adaptability
Eliminate duplicate code
Increase portability

Implications:

Variability of infrastructure and required security credentials make this approach difficult to reuse across integrated systems.

Appendix A: Application Security Guidelines

Roles

When dealing with large numbers of users, aggregate common security rights combinations into roles, which are then assigned to users.

Benefits:

Reduced security administration
Easier security audits
Increased adaptability

Implications:

Make access decisions based on user rights assigned at runtime based on role(s)
Use roles as an administrative mechanism for assigning rights

Peer Reviews and Deployment

Application code should be reviewed to ensure that "back-doors", "Trojan horses" and other breaches of security are not introduced into the system. Best practices of peer /managent review include structured code walkthroughs with developers other than and including the coder, testing including tests constructed to ensure security from "hackers", and review sign-off by reviewers, testers and the developer. Development, testing and deployment should be conducted in physically separate environments with restricted access to separate staging areas.

Applications should be compiled and moved to the staging and production environments by individuals accountable for the security and management of those environments, not the developers of the application. This reduces the risk that the source code reviewed may not be represented in the staging and production executables.

Requirements for UserId and Password

The application administrator must have a great deal of control over the UserId and password for an application. These rights should include:

1. All applications should have a UserId/password used for access to the application. This UserId and password can be identical to a higher level system UserId and password which can be passed from the operating system in those cases where the application allows for this function.

2. The application should have the ability to force a password change on some period (#days/#months).

3. The application should have the ability to force a user to change an “initial” password.

4. The application should have the ability for the user to change his/her own password.

5. The password should not be available to any administrator. Lost passwords can be reset by the administrator and then changed by the user.

6. The application should have the ability to specify a minimum and maximum password size.

7. The administrator must be able to flag a UserId as inactive while retaining it for pointer resolution.

8. UserIds should have a minimum of name, telephone number, date instituted, agency, and department.

9. The application should have the ability to specify an “Idle” disconnect time.

10. The implementation of the password must assure that it is:
    a. encrypted, or
    b. implemented via a proprietary computation in which the password is not stored but is represented by the computed value. The computed value cannot be used to re-create the password.

11. The application should include the ability to set a “retry” count so that users who fail to sign-in successfully are disabled by the system.

12. UserIds must not be shared. The UserId forms the core of our ability to audit transactions and must not be shared.

13. The design should consider the use of an “expiration” date for a given UserId. This date is useful when temporary employees or consultants are given access to the system. Know “separations of employees can also be accomplished ahead of time using this capability.

Application Audit

1. The application must audit (log file) all:
   a. Sign in/sign out
   b. Access to client records (look).
   c. Updates to client records (update/add). Depending on the system’s needs, the design should consider whether the actual changes must be saved to the audit log or whether only the fact that the data was changed is sufficient.
   d. Deletes of client records.

2. Reports must be available to analyze a given or all UserIds

Data security over and from the open Internet

1. All communication of sensitive information over the open Internet (VPN) must use encryption when client/Personal information is transmitted.

2. Sensitive data, accessible via the Internet (e.g. within a Demilitarized Zone (DMZ)), must be stored in an encrypted format. Encryption/Decryption keys and passwords must not be stored such that hackers may view them from source code or by use of decompilers.

3. Replicating data from inside the firewall to the DMZ and other special techniques need to be employed in order to avoid unauthorized access to data via the public Internet.

4. Web based applications within the state intranet need not be encrypted.

Appendix A: Application Security Guidelines

UserId rights-Application

1. A matrix of user rights should be maintained in a "class" rights structure or at the individual UserId level.

2. These rights should allow specification of rights in the following minimum hierarchy.
   a. No rights
   b. Look rights
   e. Report rights
   f. Add rights
   g. Update rights
   h. Delete rights

3. This rights hierarchy should be applied at the smallest level possible. At minimum the rights would be applied to a given screen display or category of information. At maximum they would be applied to every field. As an example, a client information system might have broad categories of maintenance, client functions, and service functions. At minimum, the rights would be applied to maintenance, client functions, and service functions. Additional granularity might be added to individual maintenance functions; client demographics, address, insurance; and service entry.

UserId rights-Reporting

1. UserId rights as described above must be incorporated in all report writers. I.e. If the user does not have reporting rights in 4.b. above, he cannot run reports. If he does not have rights to maintenance, he cannot run maintenance reports.

2. In some cases, additional rights may be incorporated in an application. As an example, a web based clinical system might allow a provider to only see his own clients for which he has provided some service in the past. In this case, a report writer must limit him to only his own clients and services.

3. Some users may need access to statistics from the system but should not see client identifiers. A facility must be in place to flag these users in the UserId definition so that selected fields can be eliminated or replaced with sequential non-meaningful values.

These rights may be implemented via:

a. "Views" of the database tailored to individual classes of users. This method requires the ability to limit a UserId in the database to given views. This approach also requires additional "rights" setup when the UserId is created.
b. Coded access when using a full programming language such as Visual Basic or Java. In this case, the programming language can access the user rights in the application and tailor the "View" to the user.
c. Extracting information to a data mart environment where the information is for a specific audience.

Appendix A: Application Security Guidelines

Exporting Data to other applications/Users

When data is exported to other applications or other users several principles must be maintained:

1. The receiving system/user must have the ability to protect the data to the same extent as the originating system. I.e. If the system contains client identifiers which are highly confidential, those identifiers cannot necessarily be given to other entities without specific client release.

2. Data transmission of sensitive data outside of the firewall must adhere to the same access rules as if it were been updated/accessed over the open Internet. I. e. It must be encrypted. This applies to the use of e-mail with enclosures. WinZip with encryption should be considered for these enclosures.

3. When possible, it is preferable to transmit client/personal data without client Identifying information. Many statistical users can meet their needs without reference to this information. In this case it may be practical to use an internal Id to identify unique clients/personnel so that unduplicated statistics may be computed.

4. Formalizing the terms of information sharing in data use agreements between participating parties is good practice.

.